CSCI 1120 (Low-Level Computing), Fall 2008

Homework 2

Assigned: September 30, 2008.

Due: October 6, 2008, at 5pm.

Credit: 20 points.

1 Reading

Be sure you have read, or at least skimmed, the readings for 9/22 and 9/29, linked from the "Lecture topics and assignments" page[1].

2 Programming Problems

Do the following programming problems. You will end up with at least one code file per problem. Submit your program source (and any other needed files) by sending mail to `bmassing@cs.trinity.edu`, with each file as an attachment. Please use a subject line that mentions the course number and the assignment (e.g., "csci 1120 homework 2"). You can develop your programs on any system that provides the needed functionality, but I will test them on one of the department's Linux machines, so you should probably make sure they work in that environment before turning them in.

1. (10 points) Write a C program to compute and print the first N Fibonacci numbers. (Recall the definition of the Fibonacci numbers: $f_0 = 1$, $f_1 = 1$, and for $n > 1$, $f_n = f_{n-1} + f_{n-2}$.) You can hardcode N, but choose a value that seems to you to be both interesting and sensible. (It should be obvious what I mean by "sensible" if you try a large value of N.)

   Sample output for $N = 6$ (which is sensible but not very interesting):

   ```
   the 0-th Fibonacci number is 1
   the 1-th Fibonacci number is 1
   the 2-th Fibonacci number is 2
   the 3-th Fibonacci number is 3
   the 4-th Fibonacci number is 5
   the 5-th Fibonacci number is 8
   ```

2. (10 points) Newton's method for computing the the square root of a non-negative number x starts with an initial guess r_0 and then repeatedly refines it using the formula

   $$r_n = (r_{n-1} + (x/r_{n-1}))/2;$$

   Repetition continues until the absolute value of $(r_n)^2$ - x is less than some specified threshold value. An easy if not necessarily optimal initial guess is just x.

[1] `http://www.cs.trinity.edu/~bmassing/Classes/CS1120_2008fall/HTML/schedule.html`

Write a C program that implements this algorithm and compares its results to those obtained with the library function `sqrt`. Since we haven't talked yet about how to read values from a human user, hard-code inputs as we did for the program to compute the roots of a quadratic equation, but show what happens for different values of x and different convergence thresholds. You might also find it interesting to print the number of iterations.

Sample output (for quickly-chosen inputs — you may want different ones):

```
square root of 0:
with newton's method (threshold 0.1):  0 (0 iterations)
using library function:  0
difference:  0

square root of -4:
unable to compute square root of negative number

square root of 4:
with newton's method (threshold 0.1):  2.00061 (3 iterations)
using library function:  2
difference:  0.000609756

square root of 2:
with newton's method (threshold 0.1):  1.41667 (2 iterations)
using library function:  1.41421
difference:  0.0024531

square root of 2:
with newton's method (threshold 0.01):  1.41667 (2 iterations)
using library function:  1.41421
difference:  0.0024531

square root of 2:
with newton's method (threshold 1e-06):  1.41421 (4 iterations)
using library function:  1.41421
difference:  1.59482e-12
```

You may find the library function `fabs` (which computes the absolute value of a `double`) useful.